Interactive Simulation in a Multi-Person Virtual World

Christopher Codella, Reza Jalili, Lawrence Koved, J. Bryan Lewis,
Daniel T. Ling, James S. Lipscomb, David A. Rabenhorst, Chu P. Wang
Veridical User Environments

Alan Norton, Paula Sweeney
Computer Animation Systems

Computer Science Department
IBM T.J. Watson Research Center
Yorktown Heights, New York 10598

Greg Turk
University of North Carolina
Chapel Hill, N.C.

Contact telephone number: (914) 784-7934
email: koved@watson.ibm.com

ABSTRACT

A multi-user Virtual World has been implemented combining a flexible-object simulator with a multisensory user interface, including hand motion and gestures, speech input and output, sound output, and 3-D stereoscopic graphics with head-motion parallax. The implementation is based on a distributed client/server architecture with a centralized Dialogue Manager. The simulator is inserted into the Virtual World as a server. A discipline for writing interaction dialogues provides a clear conceptual hierarchy and the encapsulation of state. This hierarchy facilitates the creation of alternative interaction scenarios and shared multi-user environments.

KEYWORDS: user interface management system, dialog manager, virtual worlds, virtual reality, interactive simulation.

BACKGROUND

A Virtual World is an interactive, multisensory, three-dimensional environment where human-computer interactions are based on the ways we interact with the real world [5, 7]. Virtual World software tends to be complex because it must manage multiple simultaneous devices; operate as fast as the senses of the human in the loop; incorporate heterogeneous software objects (with a range of operating speeds and hardware bases) seamlessly; and be highly flexible in construction. Our architecture takes into account the need for high performance, coordination of multiple device- and application-generated events, and the need for flexibility in configuring the Virtual World. We have described the architecture in previous papers [1, 12].

We believe that many interesting Virtual Worlds will include one or more simulations [9, 19, 22] – software components with autonomous behavior – as opposed to the simpler spatial-database traversals and preprogrammed animations often seen in today's demonstrations [4, 6, 17]. Such a simulation must be able to update its state (after being perturbed by human input) at interactive speeds. We call this a *real-time simulation*, since its results must be computed within the same elapsed real time as the human's interaction.[1] In particular, this paper will describe a Virtual World called Rubber Rocks, based on the physical modeling of flexible objects [2, 13, 20].

Our Virtual World can be thought of as a set of cooperating servers and clients communicating via asynchronous messages. The flexibility of our architecture stems from an initial design decision to separate the *style* of a Virtual World from its *content* [21]. The content defines the operational characteristics of a Virtual World independent of the user interface. In the case of Rubber Rocks, the content or application is a flexible object simulator. The style, or

user interface, is implemented by device servers and a user interface dialogue. Specialized I/O devices needed for the multisensory aspects of the Virtual World are controlled by dedicated server processes, that communicate via asynchronous messages.

The Dialogue Manager, or User Interface Management System [8, 10, 11, 14, 16, 18], provides the methods of interacting with the Virtual World. It is the client process that receives events from input device servers and application programs, and sends data and control information to application programs and output devices. It is the Dialogue Manager that maps device I/O to application parameters and results. Events received from devices and applications are processed by event handlers written in the form of rules. These rules contain embedded C code [18]. In order to achieve high flexibility, device remappability, and reusability, the rule sets should be written as independent modules, each encapsulating its own state. The rule sets are designed according to their purpose in a conceptual hierarchy [12] (see Figure 1). The *specific level* refers to the interface to the servers, such as the I/O devices and application. The *generic level* is a set of rules that merge and filter data from the specific level to derive events that are independent of particular I/O devices. At the top is the *executive level* that defines the scenario or interaction sequences in a Virtual World. These rules form a toolkit for building a variety of Virtual Worlds. We have found that this dialogue structure has successfully scaled up to our latest and most complex Virtual World incorporating real-time simulation, stereoscopic display with head motion parallax, and a shared two person environment. In addition, modifications at the executive level of Rubber Rocks allowed us to easily create different interaction scenarios.

By designing a Virtual World with such a dialogue structure – style cleanly separated from content – it is relatively easy to expand the world to include multiple users [15]. Within the hierarchy of events in the dialogues that make up the interface, a selected group of events at the generic and executive levels can be passed to other Dialogue Managers to create a cooperative multi-user Virtual World. By capturing certain events at the generic and executive levels, the participants of Rubber Rocks are able to interact with each other, as well as have independent views into the Virtual World.

RUBBER ROCKS

Rubber Rocks is a Virtual World simulating a room containing flexible objects. These objects are based on the physical modeling of point masses connected by elastic bonds. They deform and sometimes break as they collide with each other, the walls and floor, and with the users' hands. Users can create, grab, hit and shoot these simulated objects (see Figure 2). The system provides interactions via hand gestures, speech recognition, speech synthesis, and sound generation. The objects and room are displayed stereoscopically[2] with head-motion parallax[3].

Two users can simultaneously interact with Rubber Rocks. They share a single underlying simulation, the Virtual World content. However, both users have their own style components. That is, both users have their own Dialogue Manager providing independent user interfaces. These do not need to be the same. For example, one user may interact with the system using speech recognition, while the other uses pop-up menus. One user may see the objects rendered with shaded surfaces while the other sees wireframes. Each user's head-motion parallax is managed by the user's own dialogue. In addition, each user's view contains a small amount of state obtained from the other user's Dialogue Manager. For example, each user sees the other's hand.

REAL-TIME SIMULATION

The content of Rubber Rocks is a flexible object simulator [2, 13, 20] that integrates Newton's second law (F=ma) over time. Objects are modelled as a network of point masses and springs (or bonds). The dynamics of the system includes external forces such as gravity, friction and repulsive forces due to collisions. Collision forces occur when an object collides with another object, a room boundary, or one of the hands.

In Rubber Rocks, a number of simple, playful objects are defined. These vary in complexity from octahedra con-

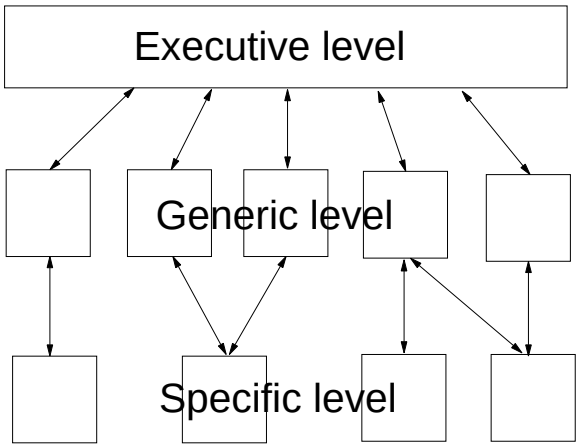

Figure 1. Hierarchical organization of rule sets

1 That is, we do not use "real-time" in its stricter technical sense here; we do not limit ourselves to simulations implemented with real-time programming principles.

2 The graphic display is computed and displayed separately for each eye.

3 The observer's viewing location is repeatedly measured, defining a new camera position from which the correct perspective view is calculated.

taining seven points and eighteen bonds, to sheets composed of 36 points and 110 bonds. Cubes, containing 27 points and 86 bonds, are objects of medium complexity. Each object has a set of physical parameters which determines the flexibility and breakability of the object. These objects can break into pieces if a strong external force is applied to the object.

User interactions with these objects result in the direct perturbation of the simulated dynamical equations, through the imposition of either forces or absolute positions. The interactions include slapping, carrying, and shooting objects[4]. Naturally, the execution rate of the simulator depends on both the complexity and the number of objects being simulated. When the simulator runs more slowly, there is a higher ratio of real time to simulated time. But worse than the loss of speed are the effects of changing the relative velocity between the objects and the user's hand.[5] For example, the user's hand will move faster with respect to the objects when the simulator is slower, and it will appear as if the objects are hit harder. Thus the ungated simulation displays both the unnatural effects of time-varying gravity, and time-varying hand-slapping forces as the contents of the room change. To simulate the flow of time more faithfully, and keep the relative ratio of hand velocity to simulated object velocity constant, we gated the simulator at a constant rate equal to the lowest expected simulation rate.

The ease of integrating the simulator shows the flexibility of our architecture. The simulator is connected into the Virtual World as a server. It communicates its state changes directly with the Dialogue Manager. The amount of state needed to be communicated to the Dialogue Man-

Figure 2. Rubber Rocks: Two users are interacting with Rubber Rocks. Each is wearing a baseball cap with a position sensor attached for head-motion parallax. Gloves and position sensors are used for gesture input. The two users are looking at a room from opposite sides. Within the room are a number of flexible objects. For this photograph, the simulator is frozen at a time when both users are tugging at a single brick-shaped object. Hands are also displayed in the scene reflecting both the position and the gesture of the users' own hands. Note that one user is viewing the scene on a twenty-three inch display while the second is using a large-screen projection system. Both views are monoscopic for purposes of the photograph.

4 Shooting is the application of a force on an object in the direction the user is pointing.

5 Velocity of the hand is computed by the tracker, while the velocity of simulated objects is computed by the simulator. The interface dialog scales the hand velocity to be in units that are proportional to the size of the simulation space.

ager is relatively small, consisting of object collisions, fractures, creations and deletions. The Dialogue Manager passes to the simulator the hand positions, forces and positions to be imposed on objects, and requests for object creation and deletion.

Graphical data is sent directly from the simulator to the renderer. The Dialogue Manager coordinates this connection by directing the simulator to pipe its output to the renderers. The graphical data is sent in the form of surfaces to describe the current shape of the flexible objects. This relatively large amount of data bypasses the Dialogue Manager for performance reasons, but also because it is not needed for computing the state of the interface dialogue.

DISPLAY METAPHOR

The display metaphor of Rubber Rocks mimics a box-shaped room located behind the display surface. In fact, the glass surface of the display forms the near wall of the room. The room is displayed stereoscopically to provide additional depth cueing. However, with traditional stereoscopic displays, the image appears to follow the observer as their head moves. But, by tracking the position of the observer's head and by using an appropriate shear[6], the room is made to appear stationary behind the screen.

The size of the room is scaled to the size of the display. When using a small 19" display, the room is really more like a box. Relatively small amounts of hand motion result in the traversal of the virtual hand across the box. Similarly, small head motions from side to side will change the view from looking down the left edge of the box to looking down the right edge. Correspondingly, with large screen projection, larger motions are needed to be consistent with having a larger virtual box behind the screen.

INTERACTION SCENARIOS: ACTIONS AND GOALS

Interaction scenarios contain two components: the possible sequences of actions the users can perform, and the goals they are trying to achieve. Consider, for example, alternative ways of controlling the camera position in Rubber Rocks. During initial debug and test, the position is controlled by filling in a form. Specific-level rule sets are written for form-widget input and events are passed on to higher-level rule sets for the control of the graphics camera position. When a 3D-tracker device becomes operational, identical events are generated by a new specific-level rule set for the tracker. This general approach allows us to substitute a variety of devices and interaction techniques without having to rewrite any of the higher level rule sets.

In a similar manner, we created scenarios with alternative goals. For general audiences, Rubber Rocks was initially demonstrated as a competitive game for multiple users. Subsequently a non-competitive version was created. In Rubber Rocks a "game" is a collection of the highest level of dialogue rules. It is generally not concerned with low-level events like hand gestures, head tracking, or graphics rendering frame rates. Sometimes, it may not even be concerned with the number of users playing the game. Rather, it is more concerned with very high-level events such as game-begin and game-end, the creation and deletion of objects, and possibly point scoring. Because the game rule sets are only concerned with this restricted set of events, we are able to quickly devise a variety of games with different goals. We can also select among these different games on the fly by enabling/disabling rules.

Of particular importance, all these alternative interactions were achieved without any changes to the simulator, the world's content.

TWO-PERSON VIRTUAL WORLD

Virtual Worlds are an ideal environment for collaboration [3]. The architecture chosen to implement Rubber Rocks has enabled us to quickly extend it into a multi-user system. The basic notion is that each user has a Dialogue Manager for their interaction with the Virtual World. Each Dialogue Manager sends commands and information to the simulator (see Real-Time Simulation). Connections just between the Dialogue Managers and the simulator would be sufficient if the participants did not need to interact with each other. To accommodate interaction between users, a connection is made between each of the Dialogue Managers. For example, each user would like to see the other's hand within the room. The hand location and posture need to be passed between Dialogue Managers. Since the Dialogue Manager we use is event-based [12], we capture events of interest and broadcast them to the other Dialogue Manager. In the case of the hand data, hand-data events are captured and broadcast to the other Dialogue Manager. Once the remote Dialogue Manager receives the hand data, it is able to update its remote-hand state information and pass the information on to the renderer for display. In addition, this information can be used to determine whether the local and remote hands are colliding. If so, the dialogue can generate events that ultimately result in hand clapping sounds. Also, if one user's dialogue is stopped, the other user's dialogue is unaffected (except that the remote hand stops moving).

Higher-level events in the dialogue structure are of most interest when designing for multi-user Virtual Worlds. In the hand example, the hand information is an aggregate of hand position and hand gesture information. Other interesting information in Rubber Rocks includes the beginning and end of a game sequence. When either user's dialogue begins or ends a game, the game begin/end event is passed

6 The view for each eye is created by virtual cameras kept parallel to each other and perpendicular to the screen.

to the other user's dialogue so it can coordinate the interface for such aspects of the interface as game scoring. More importantly, by focusing on the higher-level, and therefore more abstract, events, we need not consider the specifics of I/O devices and views into the Virtual World. For example, the abstract events are the same regardless of whether the input was spoken or chosen from a menu. Another advantage of this high-level approach to sharing of information between dialogues is the reduction of state information that needs to be communicated between Dialogue Managers.

SYSTEM CONFIGURATION

Rubber Rocks is implemented on a distributed UNIX platform. The clients and servers communicate with each other through TCP sockets and therefore Virtual Worlds can easily be implemented across heterogeneous processors [1, 12].

The distributed architecture provides the computational capacity needed for richer Virtual Worlds with complex behavior. Furthermore, the functional partitioning requires relatively low communications bandwidth. Rubber Rocks supporting two users runs on a local area network of seven IBM RISC System/6000 machines. The flexible object simulator runs on a dedicated machine due to its compute-intensive nature. Each user's interface runs on two machines – one for rendering the 3D graphics and one for running the dialogue. Two machines are dedicated to servers driving the I/O devices – gloves, 3D trackers, speech synthesizers and recognizers, and sound generators.

CONCLUSION

We built a multi-user demonstration system coupling a flexible-object simulator with a multisensory user interface, including hand motion and gestures, speech input and output, sound output, and 3-D stereoscopic graphics with head-motion parallax. The experience with the implementation of Rubber Rocks supports the basic architecture for Virtual Worlds we have developed. The flexibility of the system stems from the initial design decision to separate *style* from *content*. In addition, we developed a discipline for writing dialogues based on a conceptual hierarchy. As a result, the alternative interaction scenarios for Rubber Rocks were easily created and could be selected on the fly. This architecture also facilitated the creation of a shared, multi-user Virtual World.

We believe that Virtual Worlds rich in content will contain real-time simulations. Rubber Rocks demonstrated the integration of a simulator into our architectural framework. The simulator is inserted into the Virtual World as a server, requiring relatively little bandwidth to communicate state changes to the Dialogue Manager.

Rubber Rocks is the third in a series of increasingly complex Virtual Worlds that we have implemented on this architectural base. In the future, other simulation servers, such as constraint solvers and autonomous agents, will act as building blocks for the construction of even richer Virtual Worlds.

ACKNOWLEDGMENTS

We would like to acknowledge Wayne L. Wooten and Jeremy Stone for developing system components. In addition, we would like to thank Ron Frank for his contributions to making the Rubber Rocks demonstrations at CHI'91 and SIGGRAPH'91 possible.

References

1. Appino, Perry A., Lewis, J. Bryan, Koved, Lawrence, Ling, Daniel T., Rabenhorst, David A. and Codella, Christopher F. An Architecture for Virtual Worlds. *Presence*, 1(1), 1991.

2. Bacon, R., Gerth, J., Norton, A., Sweeney, P. and Turk, G. Tipsy Turvy. *SIGGRAPH Electronic Theater*, 1989.

3. Blanchard, C., Burgess, S., Harvill, Y., Lanier, J., Lasko, A., Oberman, M. and M. Teitel. Reality Built for Two: A Virtual Reality Tool. *Proceedings of the 1990 Symposium on Interactive 3D Graphics (Snowbird, Utah)*, 35-36, ACM, New York, 1990.

4. Brooks, F.P., Jr. Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings. *Proceedings of the 1986 Workshop on Interactive 3D Graphics (Chapel Hill, North Carolina)*, 9-21, ACM, New York, 1987.

5. Brooks, F.P., Jr. Grasping Reality Through Illusion -- Interactive Graphics Serving Science. *CHI '88 Proceedings*, 1-11, ACM, May 1988.

6. Esposito, Chris, Bricken, Meredith and Butler, Keith. Building the VSX Demonstration: Operations with Virtual Aircraft in Virtual Space. *CHI '91 Conference*, ACM, April 1991. Short Talk

7. Fisher, S.S., McGreevy, M., Humphries, J. and W. Robinett. Virtual Environment Display System. *Proceedings of the 1986 Workshop on Interactive 3-D Graphics (Chapel Hill, North Carolina)*, 1986.

8. Green, M. A Survey of Three Dialogue Models. *ACM Transactions on Graphics*, 5(3):244-275, July 1986.

9. M. Green. Virtual Reality User Interface: Tools and Techniques. In T.S. Chua and T.L. Kunii, editors, *CG International '90*, 51-68, Springer-Verlag, Tokyo, 1990.

10. Hill, R.D. Supporting Concurrency, Communication, and Synchronization in Human-Computer Interaction: The Sassafras UIMS. *ACM Transactions on Graphics*, 5(3):179-210, July 1986.

11. Jacob, R.J.K. A Specification Language for Direct-Manipulation User Interfaces. *ACM Transactions on Graphics*, 5(4):283-317, October 1986.

12. Lewis, J.B., Koved, L. and Ling, D.T. Dialogue Structures for Virtual Worlds. *CHI '91 Proceedings*, 1991.

13. Norton, A., Turk, G., Bacon, B., Gerth, J. and Sweeney, P. Animation of fracture by physical modeling. *The Visual Computer*, 7:210-219, 1991.

14. Olsen, Jr., D. MIKE: The Menu Interaction Kontrol Environment. *ACM Transactions on Graphics*, 5(4):318-344, October 1986.

15. Patterson, J. F., Hill, R. D., Rohall, S. L. and Meeks, W. S. Rendezvous: An Architecture for Synchronous Multi-User Applications. *CSCW'90 Proceedings*, 317-328, ACM, Los Angeles, CA, October 1990.

16. Pfaff, G. E., Ed. *User Interface Management Systems*. Springer-Verlag, Berlin, 1985.

17. Rheingold, H. *Virtual Reality*, chapter 8. Summit, New York, NY, 1991.

18. Rhyne, J.R. Extensions to C for Interface Programming. *Proceedings of ACM SIGGRAPH, Symposium on User Interface Software (Banff, Alberta, Canada)*, ACM, October 1988.

19. Smith, R. B. The Alternate Reality Kit: An Animated Environment for Creating Interaction Simulations. *Proceedings of the 1986 IEEE Computer Society Workshop on Visual Languages*, 99-106, 1986.

20. Sweeney, P., Norton, A., Bacon, R., Haumann, D. and Turk, G. Modelling Physical Objects for Simulation. *Proceedings Winter Simulation Conference*, Phoenix, Arizona, 1991.

21. Wiecha, C., Bennett. W., Boies, S. and Gould, J. Generating Highly Interactive User Interfaces. *CHI '89 Proceedings*, 277-282, 1989.

22. Zeltzer, D., Pieper, S. and D. Sturman. An Integrated Graphical Simulation Platform. *Proceedings of Graphics Interface '89*, 266-274, 1989.